NP-completeness of fuzzy answer set programming under Łukasiewicz semantics

Marjon Blondeel [1] and **Steven Schockaert** [2] and **Martine De Cock** [3] and **Dirk Vermeir** [4]

Abstract. Fuzzy answer set programming (FASP) is a generalization of answer set programming (ASP) in which propositions are allowed to be graded. Little is known about the computational complexity of FASP and almost no techniques are available to compute the answer sets of a FASP program. In this paper, we first present an overview of previous results on the computational complexity of FASP under Łukasiewicz semantics, after which we show NP-completeness for normal and disjunctive FASP programs. Moreover, for this type of FASP programs we will show a reduction to bilevel linear programming, thus opening the door to practical applications.

1 INTRODUCTION

Answer set programming (ASP) [1] is a form of declarative programming that can be used to model combinatorial search problems. Specifically, a search problem is translated into a disjunctive ASP program, i.e. a set of rules of the form

$$r : a_1 \vee \ldots \vee a_n \leftarrow b_1 \wedge \ldots \wedge b_m \wedge \text{not } c_1 \wedge \ldots \wedge \text{not } c_k,$$

with a_i, b_j, c_l literals (atoms or negated atoms) or constants ("true" or "false") and "not" the negation-as-failure operator. Thus, in ASP there are two types of negation: classical or strong negation "$\neg$" and negation-as-failure "not". The intuitive difference is that $\neg a$ is true when $\neg a$ can be derived, whereas $\text{not } a$ is true if a cannot be derived. Rule r indicates that whenever the body $b_1 \wedge \ldots \wedge b_m \wedge \text{not } c_1 \wedge \ldots \wedge \text{not } c_k$ holds, that the head $a_1 \vee \ldots \vee a_n$ should hold as well. For example, consider the following ASP program P.

$$\begin{array}{lllll} r_1: & \text{light} & \leftarrow & \text{power, not broken} \\ r_2: & \text{power} & \leftarrow & \overline{1} \end{array}$$

Rule r_1 informally means that we can conclude that the light is on if there is no reason to think that the lamp is broken and if we can establish that the power is on. A rule such as r_2 is called a fact; the head is unconditionally true. Given such a program, the idea is to find a minimal set of literals that can be derived from the program. These "answer sets" then correspond to the solutions of the original search problem. For example, $\{\text{light}, \text{power}\}$ is an answer set of P. Note that "power" should be an element of each answer set of P.

[1] Vrije Universiteit Brussel, Department of Computer Science, Pleinlaan 2, 1050 Brussel, Belgium, email: mblondee@vub.ac.be, Funded by a joint Research Foundation-Flanders (FWO) project
[2] Cardiff University, School of Computer Science and Informatics, 5 The Parade, Cardiff, CF24 3AA, UK, email: s.schockaert@cs.cardiff.ac.uk
[3] Ghent University, Department of Applied Mathematics and Computer Science, Krijgslaan 281, 9000 Gent, Belgium, email: martine.decock@ugent.be
[4] Vrije Universiteit Brussel, Department of Computer Science, Pleinlaan 2, 1050 Brussel, Belgium, email: dvermeir@vub.ac.be

If the head of each rule consists of exactly one literal, the program is called normal. If, in addition, a normal program does not contain "not" nor "$\neg$", it is called simple.

Given a disjunctive ASP program P and a literal l, we are interested in the following three decision problems.

1. **Existence**: Does P have an answer set?
2. **Set-membership**: Does there exist an answer set I of P such that $l \in I$?
3. **Set-entailment**: Does $l \in I$ hold for each answer set I of P?

A summary of the complexity for these decision problems is given in Table 1.

Table 1. Complexity of inference in ASP [1, 16]

	existence	set-membership	set-entailment
simple	in P	in P	in P
normal	NP-complete	NP-complete	coNP-complete
disjunctive	Σ_2^P-complete	Σ_2^P-complete	Π_2^P-complete

Recall that $\Pi_2^P = \text{co}\Sigma_2^P$, where Σ_2^P-membership means that the problem can be solved in polynomial time on a non-deterministic machine using an NP oracle.

Although ASP allows us to model combinatorial optimization problems in a concise and declarative manner, it is not directly suitable for expressing problems with continuous domains. Fuzzy answer set programming (FASP) (e.g. [20, 32]) is a generalization of ASP based on fuzzy logics [19] that is capable of modeling continuous systems by using an infinite number of truth values that correspond to intensities of properties. A (general) FASP program is set of rules of the form

$$r : g(a_1, \ldots, a_n) \leftarrow f(b_1, \ldots, b_m, \text{not } c_1, \ldots, \text{not } c_k),$$

with a_i, b_j, c_l literals (atoms or negated atoms) or constants $\overline{c}$ (with $c \in [0, 1] \cap \mathbb{Q}$) and "not" the negation-as-failure operator, and where f and g correspond to applications of fuzzy logical disjunctions and conjunctions. Rule r now intuitively means that the truth value of the head must be greater or equal than the truth value of the body. For example, consider the following program P:

$$\begin{array}{lllll} r_1: & \text{open} & \leftarrow & \text{not closed} \\ r_2: & \text{closed} & \leftarrow & \text{not open} \end{array}$$

The properties "open" and "closed" can be given a value in $[0, 1]$ depending on the extent, e.g. the angle, to which a door is opened resp. closed. The rule r_1 intuitively means that the door is open to a degree greater or equal than the extent to which the door is not closed. Rule r_2 implies the opposite property.

In recent years, a variety of approaches to FASP have been proposed (e.g. [13, 21, 23, 28, 29, 30]). The main differences are the type of connectives that are allowed, the truth lattices that are used, the definition of a model of a program and the way that partial satisfaction of rules is handled. Note that FASP is not used to deal with uncertainty, but with partial truth. See [15] for a discussion on the difference between these two concepts. To deal with uncertainty, ASP can be extended with possibility theory (e.g. [6]) or with probability theory (e.g. [2]). Still, FASP is sometimes useful as a vehicle to simulate probabilistic or possibilistic extensions of ASP, as its ability to model continuity can be used to manipulate certainty degrees [6, 14].

In particular, in this paper we are interested in disjunctive FASP programs, i.e. FASP programs with rules of the form

$$a_1 \oplus \ldots \oplus a_n \leftarrow b_1 \otimes \ldots \otimes \ldots b_m \otimes \text{not } c_1 \otimes \ldots \otimes \text{not } c_k,$$

where $\oplus$ and $\otimes$ are respectively the Łukasiewicz disjunction and the Łukasiewicz conjunction and where $\leftarrow$ is interpreted by the Łukasiewicz implicator (see Section 2.2). Other types of programs of interest are normal FASP programs, i.e. disjunctive FASP programs in which each rule has exactly one literal in the head, and simple FASP programs, i.e. normal FASP programs that do not contain "not" nor "$\neg$".

Łukasiewicz logic is often used in applications because it preserves many desirable properties from classical logic. It is closely related to mixed integer programming, as was first shown by McNaughton [24] in a non-constructive way. Later, Hähnle [18] gave a concrete, semantics-preserving, translation from a set of formulas in Łukasiewicz logic into a mixed integer program. Checking the satisfiability of a Łukasiewicz logic formula thus essentially corresponds to checking the feasibility of a mixed integer program.

By construction, FASP relates to Łukasiewicz logic as ASP does to classical logic. For Łukasiewicz logic, satisfiability is an NP-complete problem [25]. Since satisfiability has the same complexity for classical logic, one would expect ASP and FASP to have the same complexity as well. In the case of probabilistic ASP, the complexity of the existence problem has been shown to be Σ_2^P-complete [22]. On the other hand, it does not necessarily need to hold that the computational complexity remains the same, for instance in the case of description logics. There are fuzzy description logics that, unlike the classical case, do not have the finite model property under Łukasiewicz logic or under product logic [9] and there are description logics that are undecidable under Łukasiewicz logic [11]. In a previous paper [7] we showed Σ_2^P-completeness for general FASP programs under Łukasiewicz semantics for the set-membership problem "Given a program P, a value $\lambda \in [0,1] \cap \mathbb{Q}$ and a literal l. Is there an answer set I of P such that $I(l) \geq \lambda$?". However, for disjunctive FASP programs we were able to show NP-membership; a lower complexity than for the corresponding class of ASP programs. In this paper, we will extend the results from [7] by showing NP-completeness for normal and disjunctive FASP programs. Moreover, we will provide an implementation into bilevel linear programming. This result can be used as a basis to built solvers for FASP.

Intuitively, in a bilevel linear programming problem there are two agents: the leader and the follower. The leader goes first and attempts to optimize his/her objective function. The follower observes this and makes his/her decision. Since it caught the attention in the 1970s, there have been many algorithms proposed for solving bilevel linear programming problems (e.g. [4, 10, 27]). A popular way to solve such a problem, e.g. in [4], is to translate the bilevel linear programming problem into a nonlinear programming problem using Kuhn-Tucker constraints. This new program is a standard mathematical program and relatively easy to solve because all but one constraint is linear. In a later study [5], an implicit approach to satisfying the nonlinear complementary constraint was proposed, which proved to be more efficient than the known strategies.

The paper is structured as follows. In Section 2 we provide the necessary background on ASP, Łukasiewicz logic and FASP. In Section 3 we will discuss previous results about the computational complexity of FASP. In Sections 4 and 5 we will derive new complexity results for disjunctive FASP programs, and in Section 6 we provide an implementation using bilevel linear programming for this class of programs. Finally, we present some conluding remarks in Section 7.

2 BACKGROUND

2.1 Answer set programming (ASP)

A *disjunctive* ASP program is a finite set of rules of the form

$$r : a_1 \vee \ldots \vee a_n \leftarrow b_1 \wedge \ldots \wedge b_m \wedge \text{not } c_1 \wedge \ldots \wedge \text{not } c_k,$$

with a_i, b_j, c_l literals (atoms a or negated atoms $\neg a$) and/or the constants $\overline{1}$ (true) or $\overline{0}$ (false) with $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, m\}$ and $l \in \{1, \ldots, k\}$. The operator "not" is the *negation-as-failure operator*. Intuitively, the expression $\text{not } a$ is true if there is no proof that supports a. On the other hand, $\neg a$ is essentially seen as a new literal, which has no connection to a, except for the fact that answer sets containing both a and $\neg a$ will be designated as inconsistent. If l is a literal, then we define $\neg l := \neg a$ if $l = a$ with a an atom and $\neg l := a$ if $l = \neg a$ with a an atom.

We refer to the rule by its label r. The expression $a_1 \vee \ldots \vee a_n$ is called the *head* r_h of r and $b_1 \wedge \ldots \wedge b_m \wedge \text{not } c_1 \wedge \ldots \wedge \text{not } c_k$ is the *body* r_b of r. In ASP, a rule of the form "$\overline{0} \leftarrow a$", i.e. a *constraint*, is usually written as "$\leftarrow a$" and a rule of the form "$a \leftarrow \overline{1}$", i.e. a *fact*, as "$a \leftarrow$".

Different classes of ASP programs are often considered, depending on the type of rules that they contain. If P does not contain rules with negation-as-failure, it is called a *positive* (disjunctive) ASP program. If each rule in P has exactly one literal in the head, it is called a *normal* ASP program. If P is a normal ASP program that is positive, it is called a *definite* ASP program. A definite ASP program not containing strong negation is called a *simple* ASP program.

An *interpretation* I of P is any consistent set of literals $I \subseteq \mathcal{L}_P$ with

$$\mathcal{L}_P = \{l \mid l \text{ literal in } P\} \cup \{\neg l \mid l \text{ literal in } P\}$$

and where we say that I is consistent if for no literal l in $\mathcal{L}_P$ we have that $l \in I$ and $\neg l \in I$. The set of interpretations $I \subseteq \mathcal{L}_P$ will be denoted by $\mathcal{P}(\mathcal{L}_P)$. A literal l is *true w.r.t.* I, denoted as $I \models l$, if $l \in I$. An interpretation $I \in \mathcal{P}(\mathcal{L}_P)$ can be extended to rules as follows:

- $I \models \overline{1}$, $I \not\models \overline{0}$,
- $I \models \text{not } l$ iff $I \not\models l$,
- $I \models (\alpha \wedge \beta)$ iff $I \models \alpha$ and $I \models \beta$,
- $I \models (\alpha \vee \beta)$ iff $I \models \alpha$ or $I \models \beta$,
- $I \models (\alpha \leftarrow \beta)$ iff $I \models \alpha$ or $I \not\models \beta$.

with l a literal and α and β relevant expressions.

An interpretation $I \in \mathcal{P}(\mathcal{L}_P)$ is called a *model* of a disjunctive ASP program P if $I \models r$ for each rule $r \in P$. A model I of P is *minimal* if there exists no model J of P such that $J \subset I$, i.e. $J \subseteq I$ and $J \neq I$. An interpretation $I \in \mathcal{P}(\mathcal{L}_P)$ is called an *answer set* of a positive disjunctive ASP program P if it is a minimal model of P. Note that a simple ASP program P has exactly one answer set.

To define the semantics for disjunctive ASP programs P that are not positive, one starts from a candidate answer set $I \in \mathcal{P}(\mathcal{L}_P)$ and computes the Gelfond-Lifschitz reduct P^I [17] by removing all rules in P that contain expressions of the form $\text{not}\, l$ with $l \in I$ and removing all expressions of the form $\text{not}\, l$ in the remaining rules. An interpretation $I \in \mathcal{P}(\mathcal{L}_P)$ is called an answer set of P if it is an answer set of the positive program P^I.

Example 1. *Consider the normal ASP program P*

$$\begin{array}{rcl} b & \leftarrow & \text{not}\, a \\ a & \leftarrow & \text{not}\, b \end{array}$$

with a and b atoms. For an interpretation $I_1 = \{a\}$, we have that P^{I_1} is equal to

$$a \quad \leftarrow$$

Since I_1 is a minimal model of P^{I_1}, we conclude that I_1 is an answer set of P. Similar, $I_2 = \{b\}$ is also an answer set of P. One can easily check that these are the only answer sets.

Remark 1. *Note that an interpretation $I \in \mathcal{P}(\mathcal{L}_P)$ can be seen as a mapping $I : \mathcal{L}_P \to \{0, 1\}$ where $I(l) = 1$ if $l \in I$ and $I(l) = 0$ if $l \notin I$.*

Remark 2. *A disjunctive ASP program P with strong negation can be translated to a disjunctive ASP program P' without strong negation, by replacing each literal of the form $\neg a$ with a new atom a' and adding the constraint $\leftarrow a \wedge a'$. An interpretation $I \in \mathcal{P}(\mathcal{L}_P)$ is an answer set of P iff there exists an answer set $I' \in \mathcal{P}(\mathcal{L}_{P'})$ of P' such that $I(c) = I'(c)$ and $I(\neg c) = I'(c')$ for each atom $c \in \mathcal{L}_P$.*

2.2 Łukasiewicz logic

Fuzzy logics [19] form a class of logics whose semantics are based on truth degrees taken from the unit interval $[0, 1]$. Łukasiewicz logic is a particular type of fuzzy logic that is often used in applications since it preserves many properties from classical logic.

In this paper, we will consider formulas built from a set of atoms A, constants $\overline{c}$ for each element $c \in [0, 1] \cap \mathbb{Q}$ and the connectives conjunction $\otimes$, disjunction $\oplus$, negation $\sim$ and implication $\rightarrow$. The semantics of this logic are defined as follows. A *fuzzy interpretation* is a mapping $I : A \to [0, 1]$ that can be extended to arbitrary formulas as follows;

- $[\overline{c}]_I = c$,
- $[\alpha \otimes \beta]_I = \max([\alpha]_I + [\beta]_I - 1, 0)$,
- $[\alpha \oplus \beta]_I = \min([\alpha]_I + [\beta]_I, 1)$,
- $[\alpha \rightarrow \beta]_I = \min(1 - [\alpha]_I + [\beta]_I, 1)$ and
- $[\sim \alpha]_I = 1 - [\alpha]_I$.

for a constant $\overline{c}$ and α and β formulas. The set of all fuzzy interpretations $C \to [0, 1]$ with C an arbitrary set will be written as $\mathcal{F}(C)$. We say that $I \in \mathcal{F}(A)$ is a *fuzzy model* of a set of formulas B if $[\alpha]_I = 1$ for each $\alpha \in B$. For $I_1, I_2 \in \mathcal{F}(A)$ we write $I_1 \leq I_2$ if $I_1(a) \leq I_2(a)$ for each $a \in A$. A fuzzy model I of a set of formulas B is called a *minimal fuzzy model* if there does not exist a fuzzy model J of B such that $J < I$, i.e. $J \leq I$ and $J \neq I$.

2.3 Fuzzy answer set programming (FASP)

We now recall a fuzzy version of ASP based on [20], combining ASP (Section 2.1) and Łukasiewicz logic (Section 2.2).

A *general FASP program* (under Łukasiewicz semantics) is a finite set of rules of the form

$$r : g(a_1, \ldots, a_n) \leftarrow f(b_1, \ldots, b_m, \text{not}\, c_1, \ldots, \text{not}\, c_k),$$

with a_i, b_j, c_l literals (atoms a or negated atoms $\neg a$) and/or the constants $\overline{c}$ (where $c \in [0, 1] \cap \mathbb{Q}$) with $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, m\}$ and $l \in \{1, \ldots, k\}$ and "not" the negation-as-failure operator. The connectives f and g are compositions of the Łukasiewicz connectives $\otimes$ and $\oplus$. As for ASP, $\neg a$ is essentially seen as a new literal, which has no connection to a, except for the fact that answer sets containing both a and $\neg a$ "to a sufficiently high degree" will be designated as inconsistent. If l is a literal, then we define $\neg l := \neg a$ if $l = a$ with a an atom and $\neg l := a$ if $l = \neg a$ with a an atom.

We refer to the rule by its label r and $g(a_1, \ldots, a_n)$ is called the *head* r_h of r and $f(b_1, \ldots, b_m, \text{not}\, c_1, \ldots, \text{not}\, c_k)$ is called the *body* r_b of r. Rules of the form $\overline{c} \leftarrow \alpha$ with $\overline{c}$ a constant are called *constraints*. As for ASP, we will consider several classes of FASP programs. FASP programs without negation-as-failure are called *positive* FASP programs. FASP programs only containing rules of the form

$$a_1 \oplus \ldots \oplus a_n \leftarrow b_1 \otimes \ldots \otimes b_m \otimes \text{not}\, c_1 \otimes \ldots \otimes \text{not}\, c_k$$

are called *disjunctive* FASP programs. If a disjunctive FASP has exactly one literal in the head of each rule, it is called *normal* and if a normal FASP program is positive and does not contain strong negation, it is called *simple*.

A consistent *fuzzy interpretation* I of a FASP program P is any element of $\mathcal{F}(\mathcal{L}_P)$ such that $I(l) + I(\neg l) \leq 1$ for each $l \in \mathcal{L}_P$ with

$$\mathcal{L}_P = \{l \mid l \text{ literal in } P\} \cup \{\neg l \mid l \text{ literal in } P\}.$$

A fuzzy interpretation $I \in \mathcal{F}(\mathcal{L}_P)$ is extended to rules as follows:

- $[\overline{c}]_I = c$
- $[\text{not}\, l]_I = 1 - I(l)$
- $[f(\alpha, \beta)]_I = \mathtt{f}([\alpha]_I, [\beta]_I)$ where f is a prefixnotation for $\otimes$ or $\oplus$ and $\mathtt{f}$ is the corresponding function defined on $[0, 1]$ (see Section 2.2)
- $[\alpha \leftarrow \beta]_I = \min(1 - [\alpha]_I + [\beta]_I, 1)$

with $\overline{c}$ a constant, l a literal and α and β relevant expressions.

A fuzzy interpretation $I \in \mathcal{F}(\mathcal{L}_P)$ is a *fuzzy model* of a FASP program P if $[r]_I = 1$ for each rule $r \in P$. For $I_1, I_2 \in \mathcal{F}(\mathcal{L}_P)$ we write $I_1 \leq I_2$ iff $I_1(l) \leq I_2(l)$ for each $l \in \mathcal{L}_P$. A fuzzy model I of P is a *minimal fuzzy model* if there exists no model J of P such that $J < I$, i.e. $J \leq I$ and $J \neq I$. A fuzzy interpretation $I \in \mathcal{F}(\mathcal{L}_P)$ is called an *answer set* of a positive FASP program P if it is a minimal fuzzy model of P. Remark that a positive FASP program can have none, one or several answer sets [31]. However, similar as for ASP, a simple FASP program has exactly one answer set which coincides with the least fixpoint of the immediate consequence operator Π_P [13]. This operator maps fuzzy interpretations to fuzzy interpretations and is defined as

$$\Pi_P(I)(a) = \sup\{[r_b]_I \mid (a \leftarrow r_b) \in P\},$$

for an atom $a \in \mathcal{L}_P$ and $I \in \mathcal{F}(\mathcal{L}_P)$. For programs that are not positive, a generalization of the Gelfond-Lifschitz reduct [20] is used. In particular, for a program P and a fuzzy interpretation $I \in \mathcal{F}(\mathcal{L}_P)$ the reduct P^I of P w.r.t. I is obtained by replacing in each rule $r \in P$ all expressions of the form $\text{not}\, l$ by the interpretation $[\text{not}\, l]_I$; we denote the resulting rule by r^I. This new program $P^I = \{r^I \mid r \in P\}$ is a positive FASP program and I is called an answer set of P if I is an answer set of P^I.

Example 2. *Consider the normal FASP program P*

$$\begin{array}{rcl} b & \leftarrow & \text{not}\, a \\ a & \leftarrow & \text{not}\, b \end{array}$$

with a and b atoms. We show that for each $x \in [0,1]$, M_x with $M_x(a) = x$ and $M_x(b) = 1 - x$ is an answer set of P. We first compute the reduct P^{M_x}:

$$\begin{array}{rcl} b & \leftarrow & \overline{1-x} \\ a & \leftarrow & \overline{x} \end{array}$$

The minimal model of P^{M_x} is then exactly M_x. Note that there are infinitely many answer sets.

Remark 3. *Note that $[\overline{0} \leftarrow a \otimes a']_I = 1$ iff $I(a) + I(a') \leq 1$. Hence, a FASP program P with strong negation can be translated to a FASP program P' without strong negation by replacing each literal of the form $\neg a$ by a new atom a' and adding the constraint $\overline{0} \leftarrow a \otimes a'$. An interpretation $I \in \mathcal{F}(\mathcal{L}_P)$ is an answer set of a FASP program P iff there exists an answer set $I' \in \mathcal{F}(\mathcal{L}_{P'})$ of P' such that $I(c) = I'(c)$ and $I(\neg c) = I'(c')$ for each atom $c \in \mathcal{L}_P$.*

The following lemma is easily shown from the above definitions.

Lemma 1. *Let P be a FASP program such that $P = P_1 \cup C$ where C is a set of constraints in P and $I \in \mathcal{F}(\mathcal{L}_\mathcal{P})$. It holds that I is an answer set of P iff I is an answer set of P_1 and I is a fuzzy model of C.*

Remark 4. *In Lemma 1, the interpretation $I : \mathcal{L}_P \to [0,1]$ is a model of P_1 if $[r]_I = 1$ for each $r \in P_1$ and $I(l) \in [0,1]$ for $l \notin \mathcal{L}_{P_1}$.*

3 Complexity of FASP

In this section, we will recall some existing results about the computational complexity of FASP. In particular, we will consider the following decision problem. Given a general FASP program P, a literal $l \in \mathcal{L}_P$ and a value $\lambda_l \in [0,1] \cap \mathbb{Q}$, is there an answer set I of P such that $I(l) \geq \lambda_l$? We will refer to this decision problem as the *set-membership problem*.

For the computational complexity of the set-membership problem for general FASP programs, i.e. programs containing rules of the form

$$r : g(a_1, \ldots, a_n) \leftarrow f(b_1, \ldots, b_m, \text{not}\, c_1, \ldots, \text{not}\, c_k),$$

where f and g are arbitrary compositions of the Łukasiewicz connectives $\otimes$ and $\oplus$, one can show Σ_2^P-completeness. Indeed, from the complexity of fuzzy equilibrium logic [26], it follows that the set-membership problem for general FASP programs under Łukasiewicz semantics is in Σ_2^P. To show hardness, disjunctive ASP, which is Σ_2^P-hard [16] can be reduced to general FASP by replacing the classical connectives by the corresponding Łukasiewicz connectives and by adding for each literal l in P the rule $l \leftarrow l \oplus l$ to ensure that the truth value of l is either 0 or 1. In [8], for programs with exactly one atom in the head of each rule and no "$\neg$" or "not" we could only show coNP-membership. However, for some subclasses of this type of programs we could show P-membership. For example for programs having only disjunctions in the bodies of rules. We refer to [8] for an extensive overview.

Some previous results for the complexity of the set-membership problem for disjunctive FASP can be found in Table 2. In the next section we will extend these results by showing NP-completeness for normal and disjunctive FASP programs, even if constraints and strong negation are not allowed. We will also present results for other decision problems.

Table 2. Complexity of the set-membership problem for disjunctive FASP [7]

	set-membership
simple	in P
normal	in NP
disjunctive	in NP

4 NP-completeness of disjunctive FASP

In this section we will investigate the complexity of important decision problems for disjunctive FASP, i.e. the class of FASP programs that are sets of rules of the form

$$a_1 \oplus \ldots \oplus a_n \leftarrow b_1 \otimes \ldots \otimes b_m \otimes \text{not}\, c_1 \otimes \ldots \otimes \text{not}\, c_k$$

with a_i, b_j, c_l literals (atoms a or negated atoms $\neg a$) and/or the constants $\overline{c}$ (where $c \in [0,1] \cap \mathbb{Q}$) with $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, m\}$ and $l \in \{1, \ldots, k\}$.

Given a (disjunctive) FASP program P, a literal $l \in \mathcal{L}_P$ and a value $\lambda_l \in [0,1] \cap \mathbb{Q}$, we are interested in the following decision problems.

1. **Existence**: Does there exist an answer set I of P?
2. **Set-membership**: Does there exist an answer set I of P such that $I(l) \geq \lambda_l$?
3. **Set-entailment**: Is $I(l) \geq \lambda_l$ for each answer set I of P?

Remark that these decision problems are generalizations of the ones for ASP for which the complexity is given in Table 1. As we have already pointed out in the introduction, one would expect ASP and FASP to have the same computational complexity since FASP relates to Łukasiewicz logic as ASP does to classical logic and the complexity of all the main reasoning tasks in Łukasiewicz logic is as in classical propositional logic. However, as will be proved in this section, the computational complexity for disjunctive FASP turns out to be lower than the one for disjunctive ASP.

We will first show that set-membership for disjunctive FASP is NP-complete. We will do this by showing NP-membership in Proposition 1 and by showing in Proposition 2 that it is already NP-hard for a subclass of disjunctive FASP. Next, in Propositions 3 and 4, we derive resp. NP-completeness and coNP-completeness for resp. the existence and the set-entailment problem for this particular subclass. The proofs of these propositions can then be used to show resp. NP-completeness and coNP-completeness for resp. the existence and the set-entailment problem for disjunctive FASP.

Proposition 1. *Set-membership for disjunctive FASP is in NP.*

Proof. From the analysis of the geometrical structure underlying fuzzy equilibrium models [26], it follows that a FASP program P has an answer set I such that $I(l) \geq \lambda_l$ iff there is such an answer set that can be encoded using a polynomial number of bits.

Given a disjunctive program P and an answer set I; we check in polynomial time that I is an answer set of P. Note that checking if $I(l) \geq \lambda_l$ for a literal l can be done in constant time. By definition, we need to check that I is a minimal fuzzy model of P^I and that for

each $l \in \mathcal{L}_P$ we have $I(l) + I(\neg l) \leq 1$. The latter is straightforward. To check whether I is a fuzzy model of P^I, one can use linear programming. Indeed for a rule $r : a_1 \oplus \ldots \oplus a_n \leftarrow b_1 \otimes \ldots \otimes b_m$ in P^I, seen as a Łukasiewicz formula, we have that

$$\begin{aligned}
&[b_1 \otimes \ldots \otimes b_m \rightarrow a_1 \oplus \ldots \oplus a_n]_I = 1 \\
&\Leftrightarrow [(\sim b_1) \oplus \ldots \oplus (\sim b_m) \oplus a_1 \oplus \ldots \oplus a_n]_I = 1 \\
&\Leftrightarrow I(\sim b_1) + \ldots + I(\sim b_m) + I(a_1) + \ldots + I(a_n) \geq 1 \\
&\Leftrightarrow 1 - I(b_1) + \ldots + 1 - I(b_m) + I(a_1) + \ldots + I(a_n) \geq 1
\end{aligned}$$

Hence, to check whether I is a fuzzy model of P^I we use the following linear program M. The function to be minimized is the sum $\sum_{a \in P^I} a$ of all literals in P^I and the constraints in M are the following. For each literal $a \in \mathcal{L}_{P^I}$ we have $0 \leq a \leq 1$ and $a \leq I(a)$ and for each rule

$$r : a_1 \oplus \ldots \oplus a_n \leftarrow b_1 \otimes \ldots \otimes b_m$$

in P^I we have

$$1 \leq 1 - b_1 + \ldots + 1 - b_m + a_1 + \ldots + a_n$$

or equivalently

$$1 - m \leq -b_1 - \ldots - b_m + a_1 + \ldots + a_n.$$

If M has as solution $I(a)$ for each literal a, then I is a minimal fuzzy model. □

Next, to obtain NP-completeness for the set-membership problem, we prove that it is NP-hard by showing a reduction from 3SAT, which is NP-complete [12], to disjunctive FASP. 3SAT is a decision problem whose instances are Boolean expressions written in conjunctive normal form with 3 variables in each clause, i.e. expressions of the form

$$(a_{11} \vee a_{12} \vee a_{13}) \wedge (a_{21} \vee a_{22} \vee a_{23}) \wedge \ldots \wedge (a_{n1} \vee a_{n2} \vee a_{n3}),$$

where each a_{ij} is an atom or a negated atom, i.e. a literal. The problem consists of deciding whether there is a consistent assignment of "true" and "false" to the literals such that the whole expression evaluates to "true".

Proposition 2. *Set-membership for normal FASP is NP-hard if constraints are allowed.*

Proof. Consider an arbitrary instance

$$(a_{11} \vee a_{12} \vee a_{13}) \wedge (a_{21} \vee a_{22} \vee a_{23}) \wedge \ldots \wedge (a_{n1} \vee a_{n2} \vee a_{n3})$$

of 3SAT. We will refer to this expression by α. We translate each clause $a_{i1} \vee a_{i2} \vee a_{i3}$ to the rule

$$\overline{0} \leftarrow \neg a_{i1} \otimes \neg a_{i2} \otimes \neg a_{i3} \tag{1}$$

and for each literal x in α we add the rules

$$\neg x \leftarrow \text{not } x \tag{2}$$
$$x \leftarrow \text{not}(\neg x) \tag{3}$$
$$x' \leftarrow x \tag{4}$$
$$x' \leftarrow \neg x \tag{5}$$
$$\overline{0} \leftarrow \text{not}(x') \tag{6}$$

where x' is a fresh atom not used in α. We denote the resulting FASP program by P.

1. First suppose that I is an answer set of P. By Lemma 1 we know that I is an answer set of P_1 and a fuzzy model of C where P_1 is the set of all rules in P of the form (2)-(5) and C is the set of all constraints of the form (1) and (6).
   Since I is a minimal fuzzy model of $(P_1)^I$ we know that for each literal x it holds that $I(x) = 1 - I(\neg x)$ by rules (2) and (3) and $I(x') = \max(I(x), I(\neg x))$ by rules (4) and (5). Since I must be a fuzzy model of the constraints in C, it follows that $1 - I(x') = 0$ by rule (6). If $I(x') = I(x)$, then $I(x) = 1$ and $I(\neg x) = 0$. Otherwise, if $I(x') = I(\neg x)$, then $I(\neg x) = 1$ and $I(x) = 0$. Hence, I is a Boolean interpretation. Since it also holds that $I(x)+I(\neg x) \leq 1$ it is not possible that $I(x) = 1$ and $I(\neg x) = 1$. Hence I is consistent.
   Let us define the assignment G as follows. For each literal x in α we have $G(x) =$ "true" if $I(x) = 1$ and $G(x) =$ "false" if $I(x) = 0$. We check that this assignment evaluates α to "true". This follows easily by the following equations:

$$\begin{aligned}
&[\neg a_{i1} \otimes \neg a_{i2} \otimes \neg a_{i3} \rightarrow \overline{0}]_I = 1 \\
&\Leftrightarrow [\overline{0} \oplus \sim (\neg a_{i1} \otimes \neg a_{i2} \otimes \neg a_{i3})]_I = 1 \\
&\Leftrightarrow [\overline{0} \oplus \sim (\neg a_{i1}) \oplus \sim (\neg a_{i2}) \oplus \sim (\neg a_{i3})]_I = 1 \\
&\Leftrightarrow 0 + 1 - I(\neg a_{i1}) + 1 - I(\neg a_{i2}) + 1 - I(\neg a_{i3}) \geq 1
\end{aligned}$$

   Since for I it holds that $I(x) = 1 - I(\neg x)$ for each literal x, we obtain that

$$\begin{aligned}
&[\neg a_{i1} \otimes \neg a_{i2} \otimes \neg a_{i3} \rightarrow \overline{0}]_I = 1 \\
&\Leftrightarrow I(a_{i1}) + I(a_{i2}) + I(a_{i3}) \geq 1
\end{aligned}$$

   Because I is a Boolean interpretation, it must hold that $I(a_{ij}) = 1$ for at least one literal a_{ij} in each clause. Hence, G is an assignment that evaluates each clause $a_{i1} \vee a_{i2} \vee a_{i3}$, and thus the whole expression α, to "true".
2. Now suppose that P has no answer set. We need to show that there is no assignment of the literals such that expression α evaluates to "true". We will show this by contraposition.
   Consider an assignment G such that each clause $a_{i1} \vee a_{i2} \vee a_{i3}$ evaluates to "true". We define a fuzzy interpretation in $\mathcal{F}(\mathcal{L}_P)$ by $I(x) = 1$ if $G(x) =$ "true", $I(x) = 0$ if $G(x) =$ "false", $I(\neg x) = 1 - I(x)$ and $I(x') = \max(I(x), I(\neg x))$. We show that I is an answer set of P, or by Lemma 1 that it is a minimal fuzzy model of $(P_1)^I$ and a fuzzy model of C. It is clear that I is a fuzzy model of $(P_1)^I$. Now suppose there exists a fuzzy model J of $(P_1)^I$ such that $J < I$. Since I is such that $I(\neg x) + I(x) = 1$, by the rules

$$\begin{aligned}
\neg x &\leftarrow \text{not } x \\
x &\leftarrow \text{not}(\neg x)
\end{aligned}$$

   in P_1 it follows that

$$J(\neg x) \geq [\text{not } x]_I = 1 - I(x) = I(\neg x) \geq J(\neg x)$$

   and

$$J(x) \geq [\text{not}(\neg x)]_I = 1 - I(\neg x) = I(x) \geq J(x).$$

   Hence we have for each literal x that $J(x) = I(x)$ and $J(\neg x) = I(\neg x)$. Since $J < I$, there must exist a literal x such that $J(x') < I(x')$ which implies by the rules

$$\begin{aligned}
x' &\leftarrow x \\
x' &\leftarrow \neg x
\end{aligned}$$

   in P_1 that

$$I(x') > J(x') \geq I(x) \text{ and } I(x') > J(x') \geq I(\neg x).$$

This is impossible since either $I(x) = 1$ or $I(\neg x) = 1$ and then $I(x') > 1$.
It remains to be shown that I is a fuzzy model of C. Since $I(x') = \max(I(x), I(\neg x)) = 1$ we have that I models the rule $\overline{0} \leftarrow \text{not}(x')$ for each literal x. As before, we obtain

$$\begin{aligned} & [\overline{0} \leftarrow \neg(a_{i1}) \otimes \neg(a_{i2}) \otimes \neg(a_{i3})]_I = 1 \\ \Leftrightarrow \quad & I(a_{i1}) + I(a_{i2}) + I(a_{i3}) \geq 1 \end{aligned}$$

Since each clause $a_{i1} \vee a_{i2} \vee a_{i3}$ is satisfied by G, we know that for least one a_{ij} it must hold that $I(a_{ij}) = 1$. Hence $I(a_{i1}) + I(a_{i2}) + I(a_{i3}) \geq 1$.

□

Corollary 1. *Set-membership for normal FASP, if constraints are allowed, is NP-complete.*

Corollary 2. *Set-membership for disjunctive FASP is NP-complete.*

Proof. Follows by the reduction in the proof of Proposition 2. □

Proposition 3. *Existence for normal FASP, if constraints are allowed, is NP-complete.*

Proof. The same proof as for Proposition 1 can be used to show NP-membership and the proof for NP-hardness is entirely analogue to the proof for Proposition 2. □

Corollary 3. *Existence for disjunctive FASP is NP-complete*

Proof. Follows by the proof of Proposition 3. □

Proposition 4. *Set-entailment for normal FASP, if constraints are allowed, is coNP-complete.*

Proof. Let us denote normal FASP for which constraits are allowed by the term "extended normal FASP".

1. One can show that the complementary decision problem, i.e. "Given an extended normal FASP program P, a literal $l \in \mathcal{L}_P$ and a value $\lambda_l \in [0,1] \cap \mathbb{Q}$; is there an answer set I of P such that $I(l) < \lambda_l$?" is in NP by adjusting the proof of Proposition 1; it now has to be checked whether $I(l) < \lambda_l$ instead of $I(l) \geq \lambda_l$. This shows coNP-membership.
2. To show coNP-hardness, we reduce the NP-hard problem "existence" to the complement of the set-entailment problem. Consider an extended normal FASP program P. Define $P' = P \cup \{a \leftarrow a\}$ with a a fresh atom. Note that I is an answer set of P iff I' with $I'(a) = 0$ and $I'(x) = I(x)$ otherwise is an answer set I' of P'. If there exists an answer set I' of P' such that $I'(a) < 0.5$, then $I = I'_{|\mathcal{L}_P}$ is an answer set of P. If there does not exist an answer set I' of P' such that $I'(a) < 0.5$, then P has no answer sets since for each answer set I' it must hold that $I'(a) = 0$. Hence P has an answer set iff it is not the case that all answer sets I' of P' are such that $I'(a) \geq 0.5$.

□

Corollary 4. *Set-entailment for disjunctive FASP is coNP-complete.*

Proof. Follows by the proof of Proposition 4. □

A summary of these results can be found in Table 3.

5 COMPLEXITY OF DISJUNCTIVE FASP PROGRAMS WITHOUT STRONG NEGATION OR CONSTRAINTS

In this section we will investigate the complexity of the set-membership for disjunctive FASP if strong negation and constraints are not allowed and show that it remains NP-complete. Moreover, we will proof that for normal FASP, even if strong negation is not allowed, it is also NP-complete.

First, we provide a lemma that enables us to simulate constraints of a FASP program. In this lemma we will use the notation $f_{|A}$ to denote the function that is the restriction of $f : B \to C$ to the domain $A \subseteq B$.

Lemma 2. *Consider a FASP program $P = P_1 \cup C$ where P_1 is a FASP program and C is a set of constraints of the form $\overline{0} \leftarrow \alpha$. Let $P' = P_1 \cup C' \cup \{z \leftarrow \text{not}\, y\}$ where z and y are fresh atoms and $C' = \{y \leftarrow \alpha \mid (\overline{0} \leftarrow \alpha) \in C\}$.*

A fuzzy interpretation $I \in \mathcal{F}(\mathcal{L}_P)$ is an answer set of P iff there exists an answer set $I' \in \mathcal{F}(\mathcal{L}_{P'})$ such that $I'_{|\mathcal{L}_P} = I$ and $I'(z) \geq 1$.

Proof. 1. Suppose that $I \in \mathcal{F}(\mathcal{L}_P)$ is an answer set of P. Define $I' \in \mathcal{F}(\mathcal{L}_{P'})$ as $I'(a) = I(a)$ if $a \in \mathcal{L}_P$, $I'(z) = 1$ and $I'(y) = 0$. We show that I' is an answer set of P'.
First, we prove that I' is a fuzzy model of P' and thus of $(P')^{I'}$. Clearly, I' is a fuzzy model of P_1 and it models the rule $z \leftarrow \text{not}\, y$. If $y \leftarrow \alpha$ is a rule in C', then by assumption we have that $I = I'_{|\mathcal{L}_P}$ models the rule $\overline{0} \leftarrow \alpha$. Thus $[\overline{0} \leftarrow \alpha]_{I'} = 1$ and $[\alpha]_{I'} = 0 = I'(y)$. Hence I' models $y \leftarrow \alpha$.
Next, we show that I' is a minimal fuzzy model of $(P')^{I'}$. Suppose there exists a fuzzy model $J' \in \mathcal{F}(\mathcal{L}_{P'})$ of $(P')^{I'}$ such that $J' \leq I'$. We show that $J = J'_{|\mathcal{L}_P}$ is a fuzzy model of P^I. Clearly, J is a fuzzy model of $(P_1)^I$. Since $J' \leq I'$ we have that $J'(y) \leq I'(y) = 0$, thus given a rule $r : \overline{0} \leftarrow \alpha$ in C we have that for the corresponding rule $y \leftarrow \alpha$ in C' it holds that $0 = J'(y) \geq [\alpha^I]_J$, with α^I the reduct of the expression α w.r.t. I. Hence $[r^I]_J = 1$. Because I is a minimal fuzzy model of P^I, it follows that $I = J$. As mentioned before, we have $J'(y) = I'(y)$ and since $[z \leftarrow [\text{not}\, y]_{I'}]_{J'} = 1$, we also have $J'(z) \geq 1 - I'(y) = I'(z) \geq J'(z)$. Hence $I' = J'$, which shows that I' is a minimal fuzzy model of $(P')^{I'}$.
2. Suppose that $I' \in \mathcal{F}(\mathcal{L}_{P'})$ is an answer set of P' such that $I'(z) = 1$. We show that $I = I'_{|\mathcal{L}_P}$ is an answer set of P. By Lemma 1 it is sufficient to show that I is an answer set of P_1 and a fuzzy model of C.
First, we show that I is a fuzzy model of C. Since I' is a minimal fuzzy model of $(P')^{I'}$, it must hold that $I'(z) = 1 - I'(y)$ and thus that $I'(y) = 0$. Given a rule $r : \overline{0} \leftarrow \alpha$ in $\mathcal{C}$ we have that for the corresponding rule $y \leftarrow \alpha$ in C' it holds that $0 = I'(y) \geq [\alpha]_{I'}$, and thus $[r]_I = 1$.
Next, note that I is a fuzzy model of $(P_1)^I$ since I' is a fuzzy model of $(P_1)^{I'}$. Now suppose there exists a fuzzy model $J \in \mathcal{F}(\mathcal{L}_{P_1})$ of $(P_1)^I$ such that $J \leq I$. Define $J' \in \mathcal{F}(\mathcal{L}_{P'})$ as follows: $J'(a) = J(a)$ if $a \in \mathcal{L}_P$, $J'(y) = 0$ and $J'(z) = 1$. We show that J' is a fuzzy model of $(P')^{I'}$. By assumption, J' is a fuzzy model of $(P_1)^{I'}$. For the rule $r : z \leftarrow \text{not}\, y$ in P' we have $J'(z) = 1 = I'(z) \geq [\text{not}\, y]_{I'}$, hence J' models r^I. Finally, given a rule $r : y \leftarrow \alpha$ in C' we have for the corresponding rule $\overline{0} \leftarrow \alpha$ in C that $J'(y) = 0 \geq [\alpha^{I'}]_{J'}$. Hence J' models $r^{I'}$. Since $J' \leq I'$ and I' is a minimal fuzzy model of $(P')^{I'}$ it

Table 3. Complexity of inference in disjunctive FASP

	existence	set-membership	set-entailment
disjunctive FASP	NP-complete	NP-complete	coNP-complete
normal FASP, if constraints are allowed	NP-complete	NP-complete	coNP-complete

follows that $J' = I'$ and thus $J = I$.

□

We use this lemma to show a variation of the reduction proposed in the proof of Proposition 2.

Proposition 5. *Set-membership for normal FASP is NP-hard.*

Proof. Consider an arbitrary instance

$$(a_{11} \vee a_{12} \vee a_{13}) \wedge (a_{21} \vee a_{22} \vee a_{23}) \wedge \ldots \wedge (a_{n1} \vee a_{n2} \vee a_{n3})$$

of 3SAT. We will refer to this expression by α. As shown in the proof op Proposition 2, α is satisfied by an assigment G iff the propositional interpretation I, with $I(x) = 1$ if $G(x) =$ "true" and $I(x) = 0$ if $G(x) =$ "false" is an answer set of P with P the program obtained by translating each clause $a_{i1} \vee a_{i2} \vee a_{i3}$ (see the proof of Proposition 2).

By Remark 3 it follows that P can be rewritten to a disjunctive FASP P' without strong negation and in which the head contains exactly one atom or the constant $\overline{0}$ such that there is a one-on-one correspondence between the answer sets. By Lemma 2, it follows that we can define a disjunctive FASP program P'' such that the answer sets of P' correspond to the answer sets of P'' for which a certain atom has at least truth value 1. □

Corollary 5. *Set-membership for normal FASP is NP-complete, even is strong negation is not allowed.*

Proof. Follows by the reduction in the proof of Proposition 5. □

Corollary 6. *Set-membership for disjunctive FASP programs is NP-complete, even if constraints and strong negation are not allowed.*

Proof. Follows by the reduction in the proof of Proposition 5. □

A summary of these results can be found in Table 4.

6 Reduction to bilevel linear programming

In this section, we will show that we can translate disjunctive FASP programs into bilevel linear programs such that all solutions of the bilevel linear program are answer sets and if there are no solutions, then there are no answer sets. Bilevel linear programming problems are optimization problems in which the set of all variables is divided into two sets $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$. Each possibility of assignments to the variables will be denoted by the vector $\mathbf{x} = (x_1, \ldots, x_n)$ for X and by the vector $\mathbf{y} = (y_1, \ldots, y_n)$ for Y.

Intuitively, there are two agents, a leader who is responsible for the variables in X and a follower responsible for the variables in Y. Each vector $\mathbf{y}$ has to be chosen by the follower in function of the choice by the leader $\mathbf{x}$ as an optimal solution of the so-called *lower level problem* or the *follower's problem*. Knowing this reaction, the leader then wants to optimize his objective function in the so-called *upper level problem* or the *leader's problem*.

In a bilevel linear program all objective functions and constraints are linear. In particular, the type of bilevel linear programming problem in which we are interested is given by Bard [3]:

$$\begin{array}{l} \arg\min_{\mathbf{x}} c_1\mathbf{x} + d_1\mathbf{y} \\ \text{s.t. } A_1\mathbf{x} + B_1\mathbf{y} \leq b_1 \\ \qquad \arg\min_{\mathbf{y}} c_2\mathbf{x} + d_2\mathbf{y} \\ \qquad \text{s.t. } A_2\mathbf{x} + B_2\mathbf{y} \leq b_2 \end{array}$$

where $c_1, c_2 \in \mathbb{R}^n$, $d_1, d_2 \in \mathbb{R}^m$, $b_1 \in \mathbb{R}^p$, $b_2 \in \mathbb{R}^q$, $A_1 \in \mathbb{R}^{p\times n}$, $B_1 \in \mathbb{R}^{p\times m}$, $A_2 \in \mathbb{R}^{q\times n}$ and $B_2 \in \mathbb{R}^{q\times m}$.

Now consider a disjunctive FASP program P. We will translate P to a bilevel linear program Q such that the solutions of Q correspond to the answer sets of P. By definition I is an answer set of P iff I is an answer set of P^I. Informally, a guess I needs to be made first and then it has to be checked whether this guess corresponds to an answer set of P. If $\mathcal{L}_P = \{a_1, \ldots, a_n\}$, then we will define the vector $\mathbf{a} = (a_1, \ldots, a_n)$ and the vector $\mathbf{a}' = (a'_1, \ldots, a'_n)$ where each a'_i intuitively corresponds to a guess for a_i. For each such guess I, $I(a_i) = a'_i$, we want to check if it is a minimal fuzzy model of P^I. Note that P^I is a positive FASP program in which each rule is of the form

$$r : l_1 \oplus \ldots \oplus l_n \leftarrow x_1 \otimes \ldots \otimes x_m, \tag{7}$$

with l_i, x_j literals and/or constants. Similar to a previous calculation in Proposition 2, if a fuzzy interpretation $J \in \mathcal{F}(\mathcal{L}_P)$ is a model of r, then it must hold that

$$J(l_1) + \ldots + J(l_n) \geq J(x_1) + \ldots + J(x_m) - (m-1).$$

Thus for each rule $r \in P^I$ we have a constraint $x_1 + \ldots + x_m - m + 1 \leq l_1 + \ldots + l_n$.

Hence, for each guess $\mathbf{a}'$, i.e. an interpretation I, we check if there is a minimal model J of P^I such that $J(a_i) \leq I(a_i) = a'_i$ by minimizing all elements in the vector $\mathbf{a}$ subject to the constraints arising from P^I. This is the follower's problem. Finally, the guess is chosen such that the differences between $J(a_i)$ and a'_i are as small as possible. This can be done by minimizing the function $\sum_{i=1}^n (a'_i - a_i)$. If this sum is equal to 0, we have found a answer set. If this sum is not equal to 0, there cannot be an answer set.

More structured, we have

$$\begin{array}{l} \arg\min_{\mathbf{a}'} \sum_{i=1}^n (a'_i - a_i) \\ \quad \text{s.t. } \arg\min_{\mathbf{a}} \sum_{i=1}^n a_i \\ \qquad \text{s.t. } a_i + \neg a_i \leq 1, \\ \qquad 0 \leq a_i \leq 1, a_i \leq a'_i \text{ and} \\ \qquad \sum_{j=1}^m x_j - m + 1 \leq \sum_{i=1}^n l_i \text{ for each rule (7)} \\ \qquad \text{with } m, n \in \mathbb{N} \text{ in the reduct w.r.t. } \mathbf{a}' \end{array}$$

Remark 5. *A similar construction can be used if ASP is combined with other fuzzy logics, e.g. product logic, but the resulting bilevel program will not necessarily be linear.*

7 CONCLUSIONS

We have analyzed the computational complexity of FASP under Łukasiewicz semantics. In particular, when restricting to disjunctions in the head of rules and conjunctions in the bodies of rules,

Table 4. Complexity of the set-membership problem for disjunctive FASP

	set-membership
normal FASP, even if strong negation is not allowed	NP-complete
disjunctive FASP, even if constraints and strong negation are not allowed	NP-complete

i.e. disjunctive FASP programs, NP-completeness was shown, which stands in contrast with the fact that disjunctive ASP is Σ_2^P-complete. This results even holds when restricting to disjunctive FASP without strong negation and with exactly one literal in the head of each rule. Hence, allowing disjunctions in the head has no influence on the computational complexity. Given that we have not been able to show NP-membership for normal FASP programs in which both conjunction and disjunction are allowed in the bodies of rules, it is tempting to speculate that, unlike in the classical case, allowing disjunction in the body affects the computational complexity, whereas allowing it in the head does not. Finally, we have proposed an implementation of disjunctive FASP using bilevel linear programming which opens the door to practical applications.

REFERENCES

[1] C. Baral, *Knowledge Representation, Reasoning and Declarative Problem Solving*, Cambridge University Press, 2003.

[2] C. Baral, M. Gelfond, and J.N. Rushton, 'Probabilistic reasoning in computer science', in *Logic Programming and Nonmonotonic Reasoning, 7th International Conference*, pp. 21–33, (2004).

[3] J. Bard, *Practical Bilevel Optimization: Algorithms and Applications*, Kluwer Academic Publishers: USA, 1998.

[4] J. Bard and J. Falk, 'An explicit solution to the multi-level programming problem', *Computers and Operations Research*, **9**, 77–100, (1982).

[5] J. Bard and J.T. Moore, 'A branch and bound algorithm for the bilevel programming problem', *SIAM Journal on Scientific and Statistical Computation*, **11**, 281–292, (1990).

[6] K. Bauters, S. Schockaert, M. De Cock, and D. Vermeir, 'Possibilistic answer set programming revisited', in *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence* , (2010).

[7] M. Blondeel, S. Schockaert, M. De Cock, and D. Vermeir, 'Complexity of fuzzy answer set programming under Łukasiewicz semantics: first results', in *Poster Proceedings of the 5th International Conference on Scalable Uncertainty Management*, pp. 7–12, (2011).

[8] M. Blondeel, S. Schockaert, M. De Cock, and D. Vermeir, 'Fuzzy autoepistemic logic: Reflecting about knowledge of truth degrees', in *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pp. 616–627, (2011).

[9] F. Bobillo, F. Bou, and U. Straccia, 'On the failure of the finite model property in some fuzzy description logics', *Fuzzy Sets and Systems*, **172**(23), 1–12, (2011).

[10] W. Candler and R. Townsley, 'A linear two-level programming problem', *Computers and Operations Research*, **9**, 59–76, (1982).

[11] M. Cerami and U. Straccia, 'On the undecidability of fuzzy description logics with GCIs with Łukasiewicz t-norm', Technical report, (2011).

[12] S.A. Cook, 'The complexity of theorem-proving procedures', in *Proceedings of the 3rd Annual ACM Symposium on the Theory of Computing*, pp. 151–158, (1971).

[13] C.V. Damásio and L.M. Pereira, 'Antitonic Logic Programs', in *Proceedings of the 6th International Conference on Logic Programming and Nonmonotonic Reasoning*, pp. 379–392, (2001).

[14] A. Dekhtyar and V.S. Subrahmanian, 'Hybrid probabilistic programs', in *Proceedings of the 14th International Conference on Logic Programming*, pp. 391–405, (1997).

[15] D. Dubois and H. Prade, 'Possibility theory, probability theory and multiple-valued logics: a clarification', *Annals of Mathematics and Artificial Intelligence*, **32**(1-4), 35–66, (2001).

[16] T. Eiter and G. Gottlob, 'Complexity Results for Disjunctive Logic Programming and Application to Nonmonotonic Logics', in *Proceedings of the International Logic Programming Symposium*, pp. 266–278, (1993).

[17] M. Gelfond and V. Lifschitz, 'The Stable Model Semantics for Logic Programming', in *Proceedings of the Fifth International Conference and Symposium on Logic Programming*, pp. 1070–1080, (1988).

[18] R. Hähnle, 'Proof theory of many-valued logic - linear optimization - logic design: connections and interactions', *Soft Computing*, **1**, 107–119, (1997).

[19] P. Hájek, *Metamathematics of Fuzzy Logic*, Trends in Logic, 1998.

[20] J. Janssen, S. Schockaert, D. Vermeir, and M. De Cock, 'General Fuzzy Answer Set Programs', in *Proceedings of the International Workshop on Fuzzy Logic and Applications*, pp. 353–359, (2009).

[21] Y. Loyer and U. Straccia, 'Epistemic foundation of stable model semantics', *Theory and Practice of Logic Programming*, **6**(4), 355–393, (2006).

[22] T. Łukasiewicz, 'Many-valued disjunctive logic programs with probabilistic semantics', in *LPNMR*, pp. 277–289, (1999).

[23] T. Łukasiewicz and U. Straccia, 'Tightly integrated fuzzy description logic programs under the answer set semantics for the semantic web', in *Proceedings of the 1st International Conference on Web Reasoning and Rule Systems*, pp. 289–298, (2007).

[24] R. McNaughton, 'A theorem about infinite-valued sentential logic', *The Journal of Symbolic Logic*, **16**(1), (1951).

[25] D. Mundici, 'Satisfiability in many-valued sentential logic is NP-complete', *Theoretical Computer Science*, **52**(5), 145–153, (1987).

[26] S. Schockaert, J. Janssen, D. Vermeir, and M. De Cock, 'Answer sets in a fuzzy equilibrium logic', in *Proceedings of the 3rd International Conference in Web Reasoning and Rule Systems*, pp. 135–149, (2009).

[27] C. Shi, J. Lu, G. Zhang, and H. Zhou, 'An extended branch and bound algorithm for linear bilevel programming', *Applied Mathematics and Computation*, **180**(2), 529–537, (2006).

[28] U. Straccia, 'Query answering in normal logic programs under uncertainty', in *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pp. 687–700, (2005).

[29] U. Straccia, 'Annotated answer set programming', in *Proceedings of the 11th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, (2006).

[30] U. Straccia, 'Query answering under the any-world assumption for normal logic programs', in *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning*, pp. 329–339, (2006).

[31] U. Straccia, M. Ojeda-Aciego, and C. V. Damásio, 'On fixed-points of multi-valued functions on complete lattices and their application to generalized logic programs', *SIAM Journal on Computing*, (5), 1881–1911, (2009).

[32] D. Van Nieuwenborgh, M. De Cock, and D. Vermeir, 'An introduction to fuzzy answer set programming', *Annals of Mathematics and Artificial Intelligence*, **50**(3-4), 363–388, (2007).